PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-185336
(43)Date of publication of application : 16.07.1996

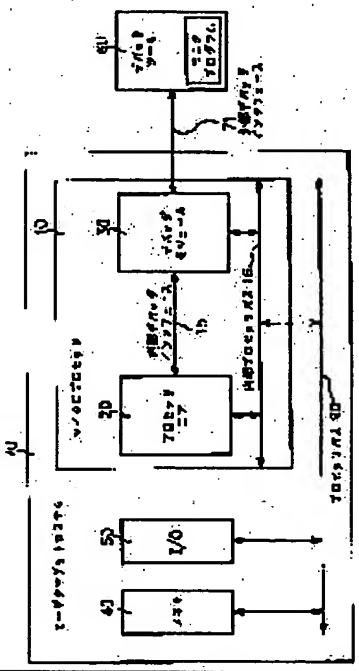
(51)Int.CI. G06F 11/22
G06F 11/28
G06F 11/28
G06F 11/28

(21)Application number : 06-339958
(22)Date of filing : 28.12.1994

(71)Applicant : HEWLETT PACKARD JAPAN LTD
TOSHIBA CORP
(72)Inventor : YANO TATSUO
MIYAMORI TAKASHI

(54) MICROPROCESSOR AND METHODS FOR TRANSMITTING AND TRACING SIGNAL BETWEEN MICROPROCESSOR AND DEBUGGING TOOL

(57)Abstract:
PURPOSE: To provide a debugging function to be sufficiently applied to a high speed processor by adding the comparatively small number of hardware parts to the microprocessor.
CONSTITUTION: A debugging module 30 taking charge of a part of a debugging function is built in the microprocessor 10 and connected to a debugging tool 60 arranged on the outside of the processor 10. In a debugging mode, a processor core 20 in the processor 10 accesses and executes a monitor program stored in the tool 60 through the module 30. During the execution of a user program by the core 20 in a normal mode, the module 30 acquires trace information, sends the information to the tool 60 and executes processing relating to a brake point.



LEGAL STATUS

[Date of request for examination] 28.12.1994
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number] 2752592
[Date of registration] 27.02.1998
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平8－185336

(43)公開日　平成８年(1996)７月16日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ	11/22	３４０　Ａ			
	11/28	Ｌ	7313－5Ｂ		
		３１０　Ａ	7313－5Ｂ		Ｃ3,Ｃ4
		３１５　Ａ	7313－5Ｂ		

審査請求　有　　請求項の数４　ＦＤ　（全 24 頁）

(21)出願番号　特願平6－339958

(22)出願日　平成６年(1994)12月28日

(71)出願人　000121914
日本ヒューレット・パッカード株式会社
東京都八王子市高倉町９番１号

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地

(72)発明者　矢野　達男
東京都八王子市高倉町９番１号　横河・ヒューレット・パッカード株式会社内

(72)発明者　宮森　高
神奈川県川崎市幸区堀川町580番１号　株式会社東芝半導体システム技術センター内

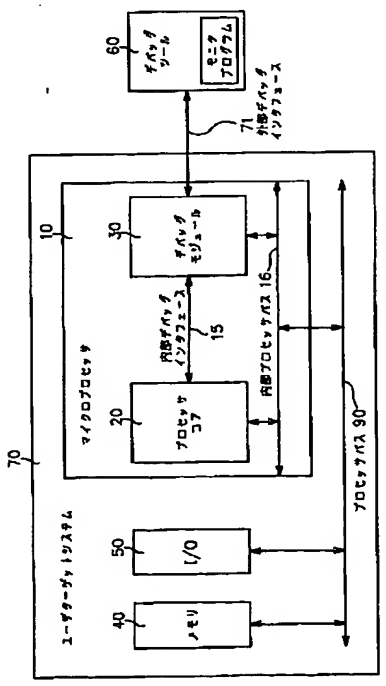
(74)代理人　弁理士　上野　英夫

(54)【発明の名称】　マイクロプロセッサ、マイクロプロセッサ－デバッグツール間信号伝送方法及びトレース方法

(57)【要約】

【目的】マイクロプロセッサに比較的少ないハードウエアを追加することにより、高速なプロセッサに対して充分なデバッグ機能を提供する。

【構成】マイクロプロセッサ１０中に、デバッグ機能の一部分を受け持つデバッグモジュール３０を設け、これをプロセッサ外のデバッグツール６０と接続する。デバッグモードにおいては、プロセッサ中のプロセッサコア２０がデバッグモジュール３０を介してデバッグツール６０内のモニタプログラムをアクセスし実行する。ノーマルモードでは、プロセッサコア２０がユーザプログラムを実行中、デバッグモジュール３０はトレース情報を取得してデバッグツールに送るとともに、ブレークポイントに関する処理を行う。



【特許請求の範囲】
【請求項１】マイクロプロセッサの応用プログラムのデバッグを行なうデバッグシステムにおいて、
前記マイクロプロセッサの外部に置かれるデバッグツール内のモニタメモリ上のモニタプログラムを動作させるための第１の論理手段、
ハードウエアブレークポイントを含むユーザプログラムの実行を制御するための第２の論理手段、及び実行されたプログラムのトレースをとるための第３の論理手段を内蔵することを特徴とするマイクロプロセッサ。
【請求項２】マイクロプロセッサがデバッグモードのとき実行されるモニタプログラムを格納するモニタメモリを設けたデバッグツールにおいて、
入力端子と出力端子及びメモリアクセス制御回路を内蔵し、
メモリリードサイクルにおいては、
前記入力端子から、前記マイクロプロセッサが直列形式で出力したアドレスを入力し、
前記メモリアクセス制御回路にて、前記直列形式のアドレスを並列形式に変換し、
前記モニタメモリからデータを読み出し、
前記メモリアクセス制御回路にて、前記並列形式で読み出されたデータを直列形式に変換し、
前記出力端子から、前記直列形式の前記データを前記マイクロプロセッサに向けて出力し、
メモリライトサイクルにおいては、
前記入力端子から、前記マイクロプロセッサが直列形式で出力したアドレス及びデータを入力し、
前記メモリアクセス制御回路にて、前記直列形式のアドレスとデータを並列形式に変換して、
前記モニタメモリ上の前記アドレスへ前記データを書き込み、
前記出力端子から、前記モニタメモリへの書き込みが終了したことを示す信号を出力することを特徴とするマイクロプロセッサーデバッグツール間信号伝送方法。
【請求項３】マイクロプロセッサが、デバッグツールとの間の信号伝送専用の信号ピンを通じて出力する命令の実行状態、及び命令がジャンプ命令または分岐命令であった場合にはジャンプ先または分岐先のアドレス、及び例外が発生した場合には当該例外の例外ベクタ番号を前記デバッグツール内のトレース用メモリへ格納することを特徴とするトレース方法。
【請求項４】マイクロプロセッサがあらかじめ設定されたアドレスの命令を実行した場合、
あらかじめ設定されたアドレスのデータをアクセスした場合、またはあらかじめ設定されたアドレスに対して、あらかじめ設定されたデータをライトした場合、
あらかじめ設定されたアドレスにから、あらかじめ設定されたデータをリードした場合、
前記マイクロプロセッサとデバッグツールとの間の信号伝送専用の信号ピンを通じて、この事象の発生を知り、トレースメモリへ格納するトレース情報の開始または終了のタイミングを決定できることを特徴とするトレース方法。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明は、マイクロコンピュータを応用したシステムのデバッグを行なうデバッグシステムに関する。
【０００２】
【従来技術】図１に従来例１を示す。従来例１は、一般にＲＯＭモニタと呼ばれるデバッグシステムである。ユーザターゲットシステム７０上に、ホストコンピュータ９０と接続するためのシリアルインターフェース８０を用意し、メモリ４０中に、モニタプログラム４１を格納しておく。マイクロプロセッサ１０は、このモニタプログラム４１を走らせることにより、Ｉ／Ｏ５０、メモリ４０、レジスタ１１にアクセスする。さらにソフトウエアブレイク命令を使用して、ユーザプログラムの実行制御を行なう。
【０００３】図２に従来例２を示す。ユーザターゲットシステム７０上のマイクロプロセッサ１０中に、デバッグツール１００との通信に必要なシリアルインターフェース１２と、デバッグツール１００から送られてきた電気信号を解釈して実行するシーケンサ１３を内蔵する。シーケンサ１３は、送られてきた信号に従って、ユーザプログラムの実行を一時停止し、レジスタ１１にアクセスしたり、バスコントローラ１４を使用して、メモリ４０やＩ／Ｏ５０にアクセスする。さらに、ハードウエアブレークポイントや、ソフトウエアブレーク命令を使用して、ユーザプログラムの実行制御を行う。シリアルインターフェース１２からの信号は、直接には、ホストコンピュータ９０に接続できないことが多いので、デバッグツール１００が、ホストコンピュータ９０からのコマンドをマイクロプロセッサ１０に接続できる電気信号に変換したり、またマイクロプロセッサ１０からの信号をホストコンピュータ９０が理解できるデータ形式に変換する。
【０００４】図３に従来例３を示す。従来例３は一般にインサーキットエミュレータと呼ばれるデバッグシステムである。デバッグ時には、ユーザターゲットボードシステム７０上のマイクロプロセッサ１０を取り外すか、または無効にし、この部分にデバッグツール１１０のプローブを接続して、デバッグ用マイクロプロセッサ１２０を代わりに動作させる。デバッグ用マイクロプロセッサ１２０は、デバッグツール上のモニタプログラムメモリ１３０に格納されているモニタプログラムを実行することにより、ユーザプログラムの実行を制御したり、メモリ４０上のデータにアクセスしたり、Ｉ／Ｏ５０をアクセスする。また、デバッグ用マイクロプロセッサ１２

０は、あたかもマイクロプロセッサ１０が実行しているかのように、ユーザターゲットシステム上のメモリ４０に格納されているプログラムを実行する。また、デバッグツール１１０は、トレースメモリ１４０を有し、デバッグ用マイクロプロセッサ１２０のプロセッサバスの状態をトレースすることができる。デバッグ用マイクロプロセッサ１１０は、マイクロプロセッサ１０からは得られないトレース情報を出力する。これにより、プロセッサバスからだけではトレースできないプロセッサの内部状態の一部がトレース可能になる。

【０００５】図４に従来例４を示す。従来例４は、一般にプリプロセッサと呼ばれているデバッグシステムである。ユーザターゲットシステム７０上のマイクロプロセッサ１０のプロセッサバス９０に、ロジックアナライザ１５０のプローブを接続することにより、マイクロプロセッサ１０のメモリ４０やＩ／Ｏ５０に対するアクセスをトレースすることができる。

【０００６】なお、図１ないし図４を用いて説明した従来例の具体的な動作や回路構成は当業者には周知であるため、これ以上詳細な説明はここでは与えない。

【０００７】

【発明が解決しようとする課題】従来例１では、モニタプログラムをユーザメモリ上で走らせるため、ユーザのユーザターゲットシステムのメモリシステムの動作が完全でない場合にモニタ自身が安定して動作しない場合があった。また、ターゲットシステムのメモリ量に余裕がない場合、モニタが占有するアドレス空間が確保できないことがあった。さらに、モニタモードへ入るためにユーザの割込みの一部を使用しなければならないので、プログラムの種類によってはデバッグが不可能なことがあった。また、あらかじめユーザターゲット上にシリアルインターフェースの回路を実装しなければならないなど、デバッグ後は使用しないかもしれない回路をターゲットシステムに設ける必要があった。また、デバッグのためのハードウエアブレークポイント等の資源をもっていないので、デバッグ機能が貧弱であり、トレースも取れなかった。

【０００８】従来例２では、シーケンサをマイクロプロセッサに内蔵し、さらに、シーケンサがレジスタにアクセスするため、デバッグツールとの接続のためのロジック回路が複雑になり、チップ上での面積も大きくなった。また、レジスタの追加などが発生した場合、シーケンサを変更しなければならないので、変更のための作業が大きかった。さらに、この従来例でもトレースは取れなかった。

【０００９】従来例３では、ユーザターゲット上のマイクロプロセッサのすべてのピンにデバッグツールを接続するため、プローブが高価になり、しかもプローブの接触が不安定になることが多かった。また、ターゲット上のメモリにアクセスする場合とデバッグツール内のモニタメモリにアクセスする場合とで、高速にバスを切替えなければならないため、動作周波数の高いプロセッサでは実現が困難であった。派生品のマイクロプロセッサがある場合には、パッケージやピンの数、位置が異なるために、本質的には同じデバッグツールが使用できるにも関わらず、プローブを派生品ひとつひとつに対して別のデバッグツールを用意しなければならなかった。また、プローブを接続することにより、ユーザターゲットで使用する信号にも影響を与え、ユーザターゲットの動作そのものを不安定にする場合があった。

【００１０】従来例４では、トレースに関しては、高速のプロセッサに対しても有効であるが、内蔵キャッシュメモリを持つプロセッサでは、キャッシュにヒットしている場合のトレースは取れなかった。また、内部にキューを持つプロセッサの場合は、フェッチされた命令が実行されたかどうかを判別できなかった。また、ユーザプログラムの実行制御機能はなく、ユーザメモリやＩ／Ｏの内容を参照したりすることはできなかった。

【００１１】

【課題を解決するための手段】従来例１の問題を解決するために、本発明では、モニタメモリをデバッグツール内に格納し、デバッグツール専用の信号を使ってモニタプログラムを動作させ、デバッグツール専用の割込みを用意し、ハードウエアブレーク機能をマイクロプロセッサに内蔵した。

【００１２】従来例２の問題を解決するために、本発明では、モニタプログラムを使用してレジスタ、ユーザメモリ、Ｉ／Ｏにアクセスする。

【００１３】従来例３の問題を解決するために、本発明では、デバッグツール内のモニタメモリでモニタプログラムを走らせるためのロジック回路、ハードウエアブレークポイントを含む実行制御を行なうためのロジック回路、及び実行された命令のＰＣ情報を出力するためのロジック回路をマイクロプロセッサに内蔵した。また、デバッグツールとの接続専用のピンをマイクロプロセッサに用意し、これらのピンだけをデバッグツールと接続した。さらにモニタプログラム実行中だけ出力されるクロックの周波数を下げる機能を付け加えた。

【００１４】従来例４の問題を解決するために、本発明では、デバッグツール内のモニタメモリでモニタプログラムを走らせるためのロジック回路、ハードウエアブレークポイントを含む実行制御を行なうためのロジック回路、及び実行された命令のＰＣ情報を出力するためのロジック回路をマイクロプロセッサに内蔵した。

【００１５】

【実施例】以下に図面を用いて本発明をアドレス３２ビット、データバス３２ビットのプロセッサコアを内蔵するマイクロプロセッサに適用した場合の実施例を示す。なお、図面及び以下の説明中、信号名に付加されたアスタリスク(*)は、その信号が負論理であることを表す。

【００１６】＜デバッグシステム全体の説明＞図５に本発明の一実施例のデバッグシステムの構成図を示す。デバッグシステムは、ユーザターゲットシステム７０とデバッグツール６０からなる。ユーザターゲットシステム７０は、デバッグ機能を内蔵したマイクロプロセッサ１０、メモリ４０、Ｉ／Ｏ５０から構成される。マイクロプロセッサ１０は、プロセッサコア２０とデバッグモジュール３０から構成される。プロセッサコア２０は、プロセッサバス２０を介して、メモリ４０やＩ／Ｏ５０をアクセスし、プログラムを実行する。また、プロセッサコア２０は、内部デバッグインターフェース１５と内部プロセッサバス１６によって、デバッグモジュール３０と接続されている。デバッグモジュール３０は、外部デバッグインターフェース７１によって、デバッグツール６０と接続されている。

【００１７】＜実行モードの説明＞デバッグシステムには、モニタプログラムを実行しているデバッグモードと、ユーザプログラムを実行しているノーマルモードの２つの実行モードがある。

【００１８】〔デバッグモードの説明〕プロセッサコア２０において、デバッグ例外、あるいはデバッグリセットが発生すると、デバッグ例外のベクタアドレス、あるいはデバッグリセットのベクタアドレスへジャンプし、デバッグモードに入る。これらのベクタアドレスに対応するメモリは、デバッグツール６０上に置かれている。プロセッサコア２０は、デバッグモジュール３０を介して、デバッグツール６０上のモニタプログラムを実行する。モニタプログラムでは、メモリ、Ｉ／Ｏ、及びレジスタのリードやライト、ハードウエアブレークポイントの設定、ユーザプログラムの実行開始アドレスの指定などの実行制御機能を実現する。プロセッサコア２０がノーマルモードへの復帰命令を実行することによって、ノーマルモードへ復帰し、ユーザプログラムの実行を開始あるいは再開する。

【００１９】〔ノーマルモードの説明〕ノーマルモードでは、デバッグシステムは、ユーザプログラムを実行する。ノーマルモードでは、外部デバッグインターフェース７１にＰＣ(program counter)情報が出力される。デバッグシステムは、ハードウエアブレーク、ソフトウエアブレーク、デバッグ割込み、デバッグリセットなどによって、プロセッサコア２０へデバッグ例外またはデバッグリセットを要求し、デバッグモードへ制御を移す。

【００２０】＜デバッグモジュールの機能の概略説明＞以下に、デバッグモードで有効なシリアルモニタバス機能と、ノーマルモードで有効な、ＰＣトレース機能、トレーストリガ機能、ハードウエアブレーク機能、ソフトウエアブレーク機能、デバッグ割込み機能、デバッグリセット機能、及びマスク機能について、説明する。

【００２１】〔シリアルモニタバス機能の説明〕シリアルモニタバス機能は、プロセッサコア２０がモニタ専用のアドレス領域をアクセスしている場合に、デバッグ専用のピンを通るシリアル伝送経路を通じてデバッグツール６０内のモニタメモリにアクセスすることによって実現される。モニタ専用メモリ以外の領域をアクセスする場合には、通常のプロセッサバスを通じたアクセスが行なわれる。これにより、モニタは、ユーザターゲットシステム上のメモリやＩ／Ｏもアクセスできる。なお、以下の実施例ではシリアルモニタバスのビット幅は１ビットであるが、マイクロプロセッサのピンをこのバスのためにもっと多く使用できる場合には、複数ビット幅にすることもできる。

【００２２】〔ＰＣトレース機能の説明〕ＰＣトレース機能は、プロセッサコア２０がユーザプログラムを実行している時に、そのプログラムカウンタＰＣの値をトレースする機能である。プロセッサコア２０がメモリ４０上のユーザプログラムを実行しているときに、そのＰＣトレース情報を内部デバッグインターフェース１５へ出力し、デバッグモジュール３０がその情報を入力して情報を加工した後、外部デバッグインターフェース７１を介して、デバッグツール６０へ出力することによって実現される。

【００２３】〔トレーストリガ機能の説明〕トレーストリガ機能には、命令アドレストレーストリガ、データアドレスブレーク、及びプロセッサバストレーストリガの３種類がある。命令アドレストレーストリガ機能は、プロセッサコア２０が内部デバッグインターフェースに出力する命令実行アドレスの値を、デバッグモジュール３０内のレジスタに設定されているアドレスの値と比較し、それらが一致した時に、トレーストリガが発生したことを外部デバッグインターフェース７１を介して、デバッグツール６０へ伝達することによって、実現される。データアドレストレーストリガ機能は、プロセッサコア２０が内部デバッグインターフェースに出力するデータアクセスアドレスの値を、デバッグモジュール３０内のレジスタに設定されているアドレスの値と比較し、それらが一致した時に、トレーストリガが発生したことを外部デバッグインターフェース７１を介してデバッグツール６０へ伝達することによって、実現される。プロセッサバストレーストリガ機能は、プロセッサコア２０がプロセッサバスに出力するデータアクセスアドレス及びそのデータの値を、デバッグモジュール３０内のレジスタに設定されているアドレス及びデータの値と比較し、それらが一致した時に、トレーストリガが発生したことを外部デバッグインターフェース７１を介してデバッグツール６０へ伝達することによって、実現される。

【００２４】〔ハードウエアブレーク機能の説明〕ハードウエアブレーク機能には、命令アドレスブレーク、データアドレスブレーク、及びプロセッサバスブレークの３種類がある。命令アドレスブレーク機能は、プロセッサコア２０が内部デバッグインターフェースに出力する

命令実行アドレスの値を、デバッグモジュール３０内のレジスタに設定されているアドレスの値と比較し、それらが一致した時に、デバッグ例外をプロセッサコア２０へ要求することによって、実現される。データアドレスブレーク機能は、プロセッサコア２０が内部デバッグインターフェースに出力するデータアクセスアドレスの値を、デバッグモジュール３０内のレジスタに設定されているアドレスの値と比較し、それらが一致した時に、デバッグ例外をプロセッサコア２０へ要求することによって、実現される。プロセッサバスブレーク機能は、プロセッサコア２０がプロセッサバスに出力するデータアクセスアドレス及びそのデータの値を、デバッグモジュール３０内のレジスタに設定されているアドレス及びデータの値と比較し、それらが一致した時に、デバッグ割込みをプロセッサコア２０へ要求することによって、実現される。

【００２５】〔ソフトウエアブレーク機能の説明〕ソフトウエアブレーク機能は、プロセッサコア２０がソフトウエアブレーク命令を実行することによりデバッグ例外を発生するものである。これにより、デバッグモードへ移行する。

【００２６】〔デバッグ割込み機能の説明〕デバッグ割込み機能は、プロセッサ２０のデバッグ割込み信号をアサートすることにより、プロセッサ２０がデバッグ例外を発生する機能である。これにより、デバッグモードへ移行する。

【００２７】〔デバッグリセット機能の説明〕デバッグリセット機能は、プロセッサ２０のデバッグリセット信号をアサートすることにより、プロセッサ２０がデバッグリセットを発生する機能である。これにより、プロセッサ２０とデバッグモジュール３０の内部状態が初期化され、プロセッサコア２０はデバッグモードとなり、デバッグリセットのベクタアドレスからプログラムの実行を開始する。

【００２８】〔マスク機能の説明〕マスク機能は、設定に従って、ノーマルモード中のユーザ割込みをマスクしたり、デバッグモード中のユーザリセットをマスクしたりする。

【００２９】＜デバッグモジュールの全体構成の詳細説明＞次にデバッグモジュール３０について、さらに詳しく説明する。

【００３０】図６にデバッグモジュール３０の内部ブロックを示す。デバッグモジュール３０には、命令／データアドレスブレーク回路３１、ＰＣトレース回路３２、プロセッサバスブレーク回路３３、シリアルモニタバス回路３４、レジスタ回路３５、外部インターフェース回路３６、分周回路３７が設けられている。

【００３１】ＰＣトレース回路３２は、プロセッサコア２０と内部デバッグインターフェース１５で接続されている。プロセッサ２０から出力される、実行された命令のＰＣ情報を入力し、その情報を加工して、外部インターフェース回路３６へ出力する。

【００３２】命令／データアドレスブレーク回路３１は、プロセッサコア２０と内部デバッグインターフェース１５で接続されている。プロセッサコア２０から出力される命令アドレスを入力し、その値がレジスタ回路３５内に設定されている命令アドレスと一致した場合、命令アドレスブレークの使用が設定されていれば、プロセッサコア２０へ命令アドレスブレーク例外を要求する。このとき、トレーストリガの使用が設定されていれば、トリガが発生したことをＰＣトレース回路３２へ通知する。

【００３３】また、プロセッサコア２０から出力されるデータアドレスを入力し、その値がレジスタ回路３５内に設定されているデータアドレスと一致した場合、データアドレスブレークの使用が設定されていれば、プロセッサコア２０へデータアドレスブレーク例外を要求する。このとき、トレーストリガの使用が設定されていれば、トリガが発生したことをＰＣトレース回路３２へ通知する。ブレークまたはトレーストリガの使用は、レジスタ回路３５内の当該レジスタの当該ビットの値によって設定される。

【００３４】プロセッサバスブレーク回路３３は、内部プロセッサバス１６を介して、プロセッサコア２０と接続されている。データは、ビット毎にマスクをすることができる。この回路は、プロセッサバス上のバスサイクルを監視し、レジスタ回路３５に設定されたアドレス及びデータと、発生したバスサイクルのアドレス及びデータが一致すると、プロセッサコア２０へ例外を要求する。このとき、トレーストリガの使用が設定されていれば、トリガが発生したことをＰＣトレース回路３２へ通知する。

【００３５】シリアルモニタバス回路３４は、内部プロセッサバス１６を介してプロセッサコア２０と接続され、プロセッサコア２０がデバッグツール６０上のモニタプログラムを実行するときに、並列形式のデータを直列形式に変換し、または直列形式のデータを並列形式に変換し、そのインターフェースを行なう。

【００３６】レジスタ回路３５は、デバッグモジュールの機能を制御する制御レジスタを含む。各レジスタ毎にアドレスが割り当てられている。内部プロセッサバス１６と内部デバッグインターフェース１５によってプロセッサコア２０と接続され、モニタプログラムを走らせることによって制御レジスタの内容をリードまたはライトすることができる。また、制御レジスタの内容は、デバッグモジュール３０内の各回路やプロセッサコア２０へ出力され、デバッグ機能が制御される。

【００３７】外部インターフェース回路３６は、デバッグモジュール３０内のＰＣトレース回路３２、シリアルモニタバス回路３４、プロセッサコア２０とデバッグツ

ール６０とのインターフェーズを制御する回路である。また、マスク機能も外部インターフェース回路３６内で実現されている。

【００３８】分周回路３７は、クロック信号ＣＬＫを分周する回路である。シリアルモニタバス回路は、分周したクロックＣＬＫ２で動作する。

【００３９】<デバッグツールの全体構成の詳細説明>図７にデバッグツールの全体構成を示す。デバッグツール６０には、通信インターフェース６２０、コントローラ６３０、モニタメモリ６４０、モニタメモリインターフェース６５０、トレースメモリインターフェース６６０、トレースメモリ６７０、ランコントロール６８０、及びターゲットインターフェース６９０が設けられている。

【００４０】通信インターフェース６２０はホストコンピュータとの通信を行なう。コントローラ６３０は、通信インターフェース６２０を経由してホストコンピュータから送られてきたコマンドを解析し、実行し、結果を送り返す。モニタメモリ６４０は、モニタプログラムを格納し、これを実行するためのメモリである。モニタメモリインターフェース６５０は、ユーザターゲットシステム７０からの直列形式の信号をモニタメモリ６４０にアクセス可能な並列形式の信号に変換し、さらにコントローラ６３０及びユーザターゲット上のマイクロプロセッサからのアクセス要求の調停を行なう。トレースメモリ６７０はユーザターゲットシステム７０上のマイクロプロセッサ１０から送られてくるＰＣ情報を格納するためのメモリである。トレースメモリインターフェース６６０は、ユーザターゲットシステム７０上のマイクロプロセッサ１０から送られてくるＰＣ情報をトレースメモリ６７０に格納する。また、コントローラ６３０からのアクセス要求があった場合に、ユーザターゲット上のマイクロプロセッサから送られてくるＰＣ情報の格納を妨げないように、これを制御する。ランコントロール６８０は、ユーザターゲットシステム７０から供給されるユーザリセット信号RESET*とユーザシステムの電源ラインの電圧VDDを入力し、デバッグ割込信号DINT*とデバッグリセット信号DRESET*をユーザターゲットシステム７０に与えることにより、ユーザプログラムをリセットしたり、停止させたり、あるいは、実行させたりする。ターゲットインターフェース６９０は、パワーオン時のユーザターゲットシステム７０及びデバッグツール６０を保護するための回路と、ターゲットユーザシステム７０の電源電圧に応じて入出力電圧を調整する回路から成る。

【００４１】<デバッグツールーマイクロプロセッサ間のインターフェース信号>デバッグツールーマイクロプロセッサ間のインターフェース信号は、全部で２０本である。内訳を以下に示す。入力／出力は、マイクロプロセッサ側から見た場合の方向を示す。

【００４２】デバッグモジュール３０とデバッグツール６０との外部デバッグインターフェース信号は、次の８本である。

1. DCLK : 出力
2. DRESET : 入力
3-5. PCST(2:0) : 出力
6. SDA0 / TPC : 出力
7. SDI / DINT : 入力
8. DBGE : 入力

【００４３】デバッグツール６０専用ではないが、下記のような信号もデバッグツール６０に接続する。

9. RESET : 出力
10. VDD : 出力

【００４４】さらに、10本のグランド線を接続する。

11-20. GND : グランド

【００４５】

(1) DCLK (Debug clock) : 出力端子

デバッグツール６０へのクロック出力である。シリアルモニタバス、ＰＣトレースインターフェースの信号のタイミングは、すべてこのデバッグクロックDCLKで規定される。シリアルモニタバス動作の時は、プロセッサコア２０の動作クロックを分周したクロックになる。

【００４６】(2) DRESET* (Debug reset) : 入力端子（プルアップ付き端子）

デバッグリセット入力。ローアクティブの信号である。DRESET*がアサートされると、ICEモジュールは初期化される（DBGEには無関係）。デバッグツール６０を使用しないときには、この端子を未接続にする。

【００４７】

(3) PCST(2:0) (PC trace status) : 出力端子

以下のPCトレースステータス情報とシリアルモニタバスのモードを出力する。以下の表に、この３ビットのPCSTの出力するコードの意味を示す。

【００４８】

【表１】

111(STL) : パイプラインストール
110(JMP) : 分岐／ジャンプ成立（PC出力あり）
101(BRT) : 分岐／ジャンプ成立（PC出力なし）
100(EXP) : 例外発生（例外ベクタコード出力あり）
011(SEQ) : シーケンシャル実行（１命令実行したことを示す）
010(TST) : パイプラインストール時のトレーストリガ出力
001(TSQ) : 実行時のトレーストリガ出力
000(DBM) : デバッグモード
(0:ローレベル、1:ハイレベル)

【００４９】(4) DBGE* (Debugger Enable) : 入力端子（プルアップ付き端子）

デバッグツール６０の接続の有無を示す。外部にデバッグツール６０が接続されていない時には、プルアップのため、ハイレベルになる。デバッグツール６０側ではロ

ーレベルにしてあるので、デバッグツールを接続するとローレベルになる。

【００５０】デバッグツール６０が接続されていない場合（DBGE*信号がハイレベルの場合）には、プロセッサコア２０のデバッグ例外ベクタアドレスはユーザに解放されている領域になり、デバッグ例外により、ユーザの作成したモニタなどへ移行することができる。また、ユーザリセットにより、デバッグモジュール機能が初期化され、デバッグ機能はハードウエアブレーク機能を除いてディスエーブルされ、マイクロプロセッサの電力消費を低減させる。また、出力信号（SDAO/TPC, DCLK, PCST[2:0]）は、すべてハイインピーダンス状態になる。

【００５１】デバッグツール６０が接続されている時には、プロセッサコア２０のデバッグ例外ベクタアドレスはユーザに解放されていないモニタ専用領域になる。この時、ユーザリセットでは、デバッグモジュールは初期化されない。これにより、ユーザリセットの直後でもデバッグ機能が活用できる。つまり、ユーザリセット直後のユーザターゲットシステムの挙動を観測したいという要求に対しても、本発明の構成によって対応することができる。

【００５２】(5) SDAO/TPC (Serial Data and address output/Target PC) ：出力端子

マイクロプロセッサがモニタプログラムを実行している時（以下、デバッグモードと称する）は、データ／アドレスをシリアルで出力する端子SDAO (Serial Data and Address Output) として、またマイクロプロセッサがユーザプログラムを実行している時（以下、ノーマルモードと称する）は、ターゲットＰＣをシリアルで出力する端子TPC (Target PC) として機能する。

【００５３】SDAOとしての動作

データ、アドレス、リードライト、バイトイネーブル信号を１ビットづつシリアルに出力する信号端子である。バスサイクル開始前にスタートビットを出力する。（つまり、１クロック間ローレベルを出力する。）出力の順は、リード時は、スタートビット（ローレベル）、A2-A19, RD, WR, BE0-BE3。ライト時は、スタートビット（ローレベル）、A2-A19, RD, WR, BE0-BE3, D0-D31である。

【００５４】TPCとしての動作

分岐／ジャンプ命令のターゲットアドレスと例外／割込みのベクタ番号を出力するための信号である。ターゲットアドレスは、下位アドレスA[2]から上位アドレスA[31]まで順に出力される。

【００５５】(6) SDI/DINT* (Serial Data Input/Debug Interrupt) ：入力端子（プルアップ付き）

デバッグモード中はシリアルデータ入力端子SDI (Serial Data input) として、またノーマルモード中はデバッグ割込端子DINT* (Debug Interrupt) として動作する。

【００５６】SDIとしての動作

データ入力信号端子。リードの時には、外部からスタートビット（ローレベル）が入力されると、次のクロックからデータ入力を開始する。ライトのとき、ローレベルが入力されると、バスサイクルが終了する。入力の順は、リード時はスタートビット（ローレベル）、D[0]-D[31]である。ライト時は終了ビット（ローレベル）だけ入力する。

【００５７】DINT*としての動作

デバッグツール６０からのデバッグ割込み入力である。デバッグツール６０を使用しない時にはこの端子を未接続にする。

【００５８】(7) RESET* (Reset) ：出力端子

ユーザリセット端子である。この信号をデバッグツールに接続することにより、例えばデバッグモジュール３０からの応答がない時にそれがユーザリセット信号によるものであるかどうかが判断できる。また、ユーザリセット直後にデバッグリセットをかけたい時には、このRESET*信号がハイレベルになった後までDRESET*を入れることで、実現が可能である。

【００５９】(8) VDD (VDD) ：出力端子

ユーザターゲットシステムの電源ラインである。これをデバッグツール６０に入力してやることにより、デバッグツール６０は、ユーザターゲットシステム７０の電源電圧を知ることができる。これにより、ユーザターゲットシステムの電源電圧に合わせて、入力波形のスレッシュホールド値を変更したり、出力波形の電圧レベルを変更したりすることができる。また、ユーザターゲットシステム７０の電源が投入されていないと判断される場合には、デバッグツール６０側の出力用デバイスをハイインピーダンスにし、デバイスの保護をはかる。

【００６０】(9) GND (GND)

デバッグツール６０間とユーザターゲットシステム７０間のグランドレベルを合わせるために、１０本のグランド線を接続する。これらは、デバッグツール６０間とユーザターゲットシステム７０間の伝送ケーブル中では上記(1)-(8)の信号の間に位置し、信号間のクロストークを低減する役割も果たす。

【００６１】＜シリアルモニタバス回路の詳細説明＞図６のシリアルモニタバス回路３４の動作について説明する。

【００６２】〔シリアルモニタバスの機能概略〕デバッグモード中に、プロセッサコア２０がモニタ専用領域をアクセスする場合、シリアルモニタバス回路３４を介して、デバッグツール６０上のメモリがアクセスされる。

【００６３】シリアルモニタバスを用いたライトオペレーションでは、シルアルモニタバス回路３４は、アドレス、バス制御信号、データをSDAO信号に、１ビットづつシリアルに出力する。リードオペレーションでは、アドレス、バス制御信号を１ビットづつシリアルに出力する。リードオペレーションでは、アドレス、バス制御信号をSDAO信号に出力し、SDI信号からデータを１ビット

づつシリアルに入力する。

【００６４】シリアルモニタバスは、プロセッサコア２０の動作クロックＣＬＫを分周したクロックＣＬＫ２で動作する。

【００６５】シリアルモニタバスに出力するアドレスはプロセッサコアのアドレス信号のA[19:2]の１８ビットであり、１Mバイトのメモリ空間をアクセスできる。

【００６６】プロセッサコア２０のバイトイネーブル信号BE[3:0]をシリアルモニタバスへ出力するので、バイト、ハーフワード、３バイトのアクセスも可能である。ただし、シリアルモニタバスでは、バイト、ハーフワード、３バイトアクセスの場合も、３２ビット分のデータをを転送する。バイト、ハーフワード、３バイトのライトのとき、BE[3:0]*がインアクティブなバイト位置のデータは不定である。リードの時は、インアクティブバイトなバイト位置のデータはプロセッサコア２０で無視され、リードされない。

【００６７】ノーマルモード中は、モニタ専用領域へのライトは無視され、リードは結果が不定となる。このようなライトアクセスがあった時、シリアルモニタバス回路３４は、プロセッサコア２０へ、バスオペレーションの完了を示すアクノリッジ信号を送り、バスオペレーションを終了する。

【００６８】＜シリアルモニタバスの信号伝送方法の詳細説明＞図８にシリアルモニタバスに関わる回路の概略図を示す。これを用いて、信号伝送の手順を説明する。

【００６９】〔プロセッサコア２０がメモリリードを行なう場合〕

（１）プロセッサコア２０から出力される並列形式のアドレス、リード信号、バイトイネーブルをシリアル出力回路Ａ３４２が直列形式に変換して、SDAOから出力する。

（２）デバッグツール６０内のシリアルモニタ入力回路Ｂ６５１はこれを入力し、並列形式に変換し、モニタメモリ６４０に出力する。

（３）メモリから出力された並列形式のデータは、シリアル出力回路Ｂ６５２で直列形式に変換され、SDIを通じて出力される。

（４）デバッグモジュール３０内のシリアル入力回路Ａ３４３はこれを並列形式に変換して、プロセッサコア２０へ出力する。

（５）プロセッサコア２０がその並列形式のデータをリードする。

【００７０】〔プロセッサコア２０がメモリライトを行なう場合〕

（１）プロセッサコア２０から出力される並列形式のアドレス、ライト信号、バイトイネーブル、データをシリアル出力回路Ａ３４２が直列形式に変換して、SDAOから出力する。

（２）デバッグツール６０内のシリアル入力回路Ｂ６５１はこれを入力し、並列形式に変換し、モニタメモリ６４０に出力する。

（３）モニタメモリ６４０への書き込みが終了したら、シリアル出力回路Ｂ６５２で、SDIに１クロック間だけローレベルを出力する。

（４）ユーザターゲットシステム７０内のマイクロプロセッサ１０中のシリアル入力回路Ａ３４３は、このローレベルを入力したら、プロセッサコア２０へライトサイクル終了したことを伝える。

（５）プロセッサコア２０は、ライトサイクルを終了する。

【００７１】＜シリアルモニタバスオペレーションのタイミングの詳細説明＞以下にシリアルモニタバスオペレーションをタイミングチャートを用いて詳しく説明する。

【００７２】〔シリアルモニタバスのリードバスオペレーション〕図９にシリアルモニタバスのリードバスオペレーションのタイミングチャートを示す。

（１）プロセッサコア２０がモニタ専用領域に対するリードバスオペレーションを開始する（サイクル１）。プロセッサコア２０は、プロセッサバスにアクセスするアドレスを並列形式で出力し、リード信号をアサートし、またリードするバイト位置のバイトイネーブル信号をアサートする。

（２）シリアルモニタバス回路３４は、モニタ領域へのリードバスオペレーションの開始を認識すると、SDAO信号にコアクロックＣＬＫを分周したＣＬＫ２信号の１クロック間ローレベルを出力する（サイクル２）。

（３）シリアルモニタバス回路３４は、プロセッサコア２０のリードバスオペレーションで出力された、アドレスA[2]-A[19]、ハイレベル（リードを示す）及びバイトイネーブル信号BE[0]*-BE[3]*を、この順でSDAO信号にＣＬＫ２信号の１クロック間づつ出力する（サイクル３～２５）。

（４）デバッグツール６０内のモニタメモリインターフェース６５０は、SDAO信号に出力されたアドレスA[2]-A[19]、ハイレベル（リードを示す）及びバイトイネーブル信号BE[0]*-BE[3]*を、この順で、DCLKに従って１クロックづつ入力し、アドレス及びバイトイネーブル信号を並列形式に変換し、モニタメモリ６４０に対して出力する。

（５）モニタメモリインターフェース６５０は、モニタメモリ６４０から出力された並列形式のデータを直列形式に変換する。データを外部へ出力する前に、１クロック間SDI信号にローレベルを出力する（サイクルｎ）。続けて、DCLKに同期して、データをD[0]から順に１ビットづつD[31]まで出力する（サイクルｎ＋１～ｎ＋３２）。

（６）シリアルモニタバス回路３４は、SDIにローレベルを検出すると（サイクルｎ）、次のサイクルよりデー

タD[0]-D[31]をDCLKの１クロック毎に読み込む（サイクルｎ＋１～ｎ＋３２）。
（７）シリアルモニタバス回路３４は、プロセッサコア２０のリードバスの応答信号をアサートし、読み込んだ３２ビットのデータをプロセッサバスへ並列形式にして出力する（サイクルｎ＋３３）。
（８）プロセッサコア２０は、プロセッサバス上のデータを読み込み、リードバスオペレーションを終了する。
【００７３】〔シリアルモニタバスのライトバスオペレーション〕図１０にシリアルモニタバスのライトバスオペレーションのタイミングチャートを示す。
（１）プロセッサコア２０がモニタ専用領域に対するライトバスオペレーションを開始する（サイクル１）。プロセッサコア２０は、プロセッサバスにアクセスするアドレスを出力し、ライト信号をアサートする。ライトするバイトの位置のバイトイネーブル信号をアサートする。
（２）シリアルモニタバス回路３４は、モニタ領域へのライトバスオペレーションの開始を認識すると、SDAO信号にコアクロックＣＬＫを分周したＣＬＫ２信号の１クロック間ローレベルを出力する（サイクル２）。
（３）シリアルモニタバス回路３４はプロセッサコア２０のライトバスオペレーションで出力された、アドレスA[2]-A[19]、ローレベル（ライトを示す）、バイトイネーブル信号BE[0]*-BE[3]*及びライトデータDOUT[0]-DOUT[31]を、この順でSDAO信号にＣＬＫ２信号の１クロック間づつ出力する（サイクル３～５７）。
（４）デバッグツール６０内のモニタメモリインターフェース６５０は、SDAO信号に出力されたアドレスA[2]-A[19]、ハイレベル（リードを示す）、バイトイネーブル信号BE[0]*-BE[3]*及びライトデータDOUT[0]-DOUT[31]をこの順でDCLKに従って、１クロックづつ入力し、アドレス、バイトイネーブル信号、ライトデータを並列形式に変換し、モニタメモリ６４０に対して出力する。
（５）モニタメモリインターフェース６５０は、モニタメモリ６４０への書き込みが終了したら、１クロック間SDI信号にローレベルを出力する（サイクルｎ）。
（６）シリアルモニタバス回路３４は、SDIにローレベルを検出すると、プロセッサコア２０に対して、ライトバス応答信号をアサートする（サイクルｎ＋１）。
（７）プロセッサコア２０は、ライトバスオペレーションを終了する。
【００７４】＜ＰＣトレース回路＞ここで、インダイレクトジャンプ、ダイレクトジャンプ、分岐を以下のように、定義する。
・インダイレクトジャンプ
レジスタに格納されているアドレスへのジャンプなど、ジャンプ先がその命令自体では特定できないジャンプ。
・ダイレクトジャンプ
命令自身が格納されているアドレス及び命令コードによってジャンプ先のアドレスが特定できるジャンプ。
・分岐
命令自身が格納されているアドレス及び命令コードの一部の和で分岐先が特定できるジャンプ。分岐において、実際にジャンプが発生するかどうかは条件により決まる。実際にジャンプが行なわれることを分岐成立、行なわれなかったことを分岐不成立と呼ぶ。
【００７５】また、ＰＣトレースには、次の２種類のトレースモードがある。
・リアルタイムトレースモード
このモードでは、プロセッサコア２０の実行は常にリアルタイムで行なわれるが、インダイレクトジャンプのターゲットＰＣ出力中に次のインダイレクトジャンプが発生した場合、先に発生したインダイレクトジャンプのターゲットＰＣの出力は中止され、新しいターゲットＰＣの出力が開始される。
・非リアルタイムトレースモード
このモードでは、上記のようにインダイレクトジャンプが接近して発生した場合、先に発生したインダイレクトジャンプのターゲットＰＣを出力し終るまでプロセッサコア２０のパイプライン処理を停止させる。これにより、プロセッサコア２０の実行のリアルタイム性は損なわれるが、インダイレクトジャンプのターゲットＰＣは必ず完全に出力される。
【００７６】ＰＣトレース回路３２は、プロセッサコア２０から以下の信号を入力する。
・デバッグモード信号
プロセッサコア２０がデバッグモード中か、ノーマルモード中かを示す。
・パイプライン実行信号
命令が一命令実行されたことを示す。
・３０ビットのターゲットPC[31:0]信号
分岐命令、ジャンプ命令のターゲットアドレス、あるいは例外のベクタアドレスを示す。以下のインダイレクトジャンプ信号、ダイレクトジャンプ命令、分岐成立信号、例外発生信号がアサートされているときに有効である。
・インダイレクトジャンプ信号
インダイレクトジャンプが実行されたことを示す。
・ダイレクトジャンプ信号
ダイレクトジャンプが実行されたことを示す。
・分岐成立信号
分岐成立の分岐命令が実行されたことを示す。
・例外発生信号
例外が発生したことを示す。
【００７７】また、ＰＣトレース回路３２は、ＰＣトレースを完全に行なうために、以下の信号をプロセッサコア２０へ出力する。
・パイプラインストール要求信号
ターゲットＰＣの出力を完全に行なう非リアルタイムト

レースモードの時、プロセッサコア２０のパイプラインをストールさせるための信号である。ＰＣトレース回路３２は、インダイレクトジャンプのターゲットＰＣを出力している時に次のインダイレクトジャンプが発生すると、この信号をアサートしてプロセッサコア２０のパイプラインをストールさせる。出力中のターゲットＰＣが完全に出力されるとこの信号をネゲートして、プロセッサコア２０のパイプライン処理を再開させる。

【００７８】ＰＣトレース回路３２は、命令／データアドレスブレーク回路３１とプロセッサブレーク回路３３からトリガ要求信号を入力する。また、レジスタ回路３５内のレジスタに割り当てられているトレースモードを切替えるビットの状態を入力する。

【００７９】ＰＣトレース回路３２は、ノーマルモード時にプロセッサコア２０が出力するＰＣトレース情報を、１ビットのＰＣ出力（TPC信号）と、３ビットのステータス信号（PCST[2:0]信号）にしてデバッグツール６０へ出力する。以下にPCST[2:0]とTPC信号について説明する。

【００８０】〔PCST[2:0]〕各クロック毎に、命令の実行状態をPCST[2:0]に出力する。下記の説明では、"0"はローレベルを、"1"はハイレベルを表す。

・111(STL):パイプラインストール
トレーストリガ出力のない状態で、命令の実行終了がなかったことを示す。

・110(JMP):分岐／ジャンプ成立（ＰＣ出力あり）
分岐成立の分岐命令、ジャンプ命令の実行が終了し、TPC信号にそのターゲットアドレス（分岐、ジャンプ先のアドレス）の出力を開始したことを示す。

・101(BRT):分岐／ジャンプ成立（ＰＣ出力なし）
分岐成立の分岐命令、ジャンプ命令の実行が終了したが、TPC信号にそのターゲットアドレス（分岐、ジャンプ先のアドレス）の出力がないことを示す。

・100(EXP):例外発生（例外ベクタのコード出力あり）
例外が発生したことを示す。同時にTPC信号へ例外ベクタのコード出力が開始されたことを示す。コードは、３ビットあり、下位code(0)からcode(1)、code(2)の順でTPC信号へ出力する。

【００８１】

例外の種類	ベクタアドレス	code	
リセット,Nmi	BFC0_0000	(100)	4
UTLB(BEV=0)	8000_0000	(000)	0
UTLB(BEV=1)	BFC0_0100	(110)	6
その他(BEV=0)	8000_0080	(001)	1
その他(BEV=1)	BFC0_0180	(111)	7

ここで、BEVはレジスタ回路３５内のあるレジスタ内の１ビットであり、その値によって、例外処理のベクタアドレスを変更できる。

【００８２】・011(SEQ):シーケンシャル実行（１命令実行したことを示す）
ジャンプ、分岐の実行(JMP,BRT)以外、あるいはトレーストリガ出力要求(TSQ)のない状態で、命令が実行されたことを示す。分岐不成立の分岐命令が実行されたときもこのコードが出力される。

・010(TST):パイプラインストール時のトレーストリガ出力
命令の実行終了がないクロックで、命令アドレストレーストリガまたは、プロセッサバストレーストリガが発生したことを示す。

・001(TSQ):実行時のトレーストリガ出力
命令の実行が終了したクロックで、命令アドレストレーストリガあるいはプロセッサバストレーストリガが発生したことを示す。

・000(DBM):デバッグモード
ノーマルモードでは、このコードが出力されることはない。

〔TPC〕分岐命令、ジャンプ命令のターゲットアドレスを出力する信号である。PCST[2:0]に110(JMP)が出力されたクロックから、ターゲットアドレスの出力を開始する。ターゲットアドレスは、下位A(2)から１クロック毎に出力される。PCST[2:0]に、100(EXP)が出力されたクロックから、例外ベクタの３ビットコードを出力する。コードは、下位code(0)から１クロック毎に出力する。

【００８３】TPC信号によって、１ビットシリアルでターゲットアドレスを出力するため、前の分岐ジャンプ命令をTPC信号へ出力している途中に、次の分岐、ジャンプ命令あるいは例外が発生することがある。この場合のTPCへのターゲットアドレス出力の優先度について、以下のように規定する。

【００８４】（１）トレースモードがリアルタイムモードである場合、ターゲットＰＣ出力中に新たにインダイレクトジャンプが発生すると、その時のターゲットＰＣ出力を終了し、必ず新たなインダイレクトジャンプのターゲットＰＣの出力を開始する。

（２）トレースモードが非リアルタイムモードである場合、ターゲットＰＣ出力中に新たにインダイレクトジャンプが発生すると、その時のターゲットＰＣ出力を終了するまでプロセッサコアのパイプライン処理を停止し、必ずそのターゲットＰＣ出力を終了してからプロセッサコアのパイプライン処理を再開し、新たなインダイレクトジャンプのターゲットＰＣの出力を開始する。

（３）ターゲットＰＣ出力中に例外が発生したときは、例外のベクタ番号（３ビット）を必ず出力し、その後、中断していたＰＣ出力を再開する。

（４）ターゲットＰＣ出力中に、新たにダイレクトジャンプ、分岐が発生したときは、そのダイレクトジャンプ、分岐のターゲットアドレスの出力は行なわない。ダイレクトジャンプ、分岐については、発生時にターゲットＰＣを出力していない時のみ、そのターゲットＰＣが出力される。ダイレクトジャンプまたは分岐の場合に

は、ターゲットアドレスが出力されなくても、その命令がアドレス何番地にストアされているのかがわかれば、メモリにストアされているその命令のコードを参照することにより、ジャンプ先、分岐先のアドレスが判明する。当該命令がアドレス何番地にストアされているかは、その前に発生したダイレクトジャンプまたは分岐から何クロック目に実行されているかで判明する。

【００８５】次にＰＣトレースの出力例を図を用いて説明する。

【００８６】（例１）分岐命令のＰＣトレース

図１１に分岐命令のＰＣトレース出力の例を示す。最初の分岐成立の分岐命令beqでは、TPCにターゲットＰＣを出力していないので、PCST[2:0]にJMPコードを出力し、TPCにターゲットPCの出力を開始する。分岐不成立の分岐命令bneでは、PCST[2:0]にSEQコードを出力する。２番目の分岐成立の分岐命令bneはダイレクトジャンプであり、最初の分岐命令のターゲットＰＣを出力しているので、TPCにターゲットPCを出力しない。PCST[2:0]には、BRTコードを出力する。

【００８７】（例２）インダイレクトジャンプ命令のＰＣトレース

図１２にインダイレクトジャンプ命令のＰＣトレース出力の例を示す。最初のインダイレクトジャンプ命令jrlでは、PCST[2:0]にJMPコードを出力し、TPCにターゲットＰＣの出力を開始する。分岐不成立の分岐命令bneでは、PCST[2:0]にSEQコードを出力する。２番目のインダイレクトジャンプ命令jr2では、最初のインダイレクトジャンプ命令のターゲットＰＣの出力を中止して、jr2のターゲットＰＣをTPCへ出力する。PCST[2:0]にはJMPコードを出力する。

【００８８】（例３）例外とインダイレクトジャンプ命令のＰＣトレース

図１３に例外とインダイレクトジャンプ命令のＰＣトレース出力の例を示す。ソフトウエアブレーク命令break例外が発生すると、PCST[2:0]にEXPコードを出力し、TPCに例外ベクタコードの出力を開始する。分岐不成立の分岐命令bneでは、PCST[2:0]にSEQコードを出力する。インダイレクトジャンプ命令jr2では、jr2のターゲットＰＣをTPCへ出力し、PCST[2:0]には、JMPコードを出力する。

【００８９】（例４）デバッグ例外発生時にＰＣを出力していない場合のＰＣトレース

図１４にデバッグ例外発生時のPCST[2:0]の出力タイミングの例を示す。同図において、DM信号は、ハイレベルでデバッグモード中であることを示し、ローレベルでノーマルモード中であることを示す、プロセッサコア２０中の内部信号である。プロセッサコア２０がデバッグ例外あるいはデバッグリセットを発生すると、デバッグモードに入る。このとき、ＰＣトレース回路３２はPCST[2:0]出力にDBMコードを出力する。ターゲットＰＣを出力していないときは、デバッグ例外を発生した命令の実行終了直後にデバッグモードになる。デバッグ例外を発生した直前の命令までのＰＣトレース情報が出力される。

【００９０】（例５）デバッグ例外発生時にＰＣを出力している場合のＰＣトレース

図１５にデバッグ例外発生時に、ターゲットＰＣを出力している場合のPCST[2:0]出力タイミングを示す。ターゲットＰＣを出力している時は、そのターゲットＰＣが完了した後、デバッグモードになる。デバッグ例外を発生した直前の命令までのＰＣトレース情報が出力される。ターゲットＰＣを出力している間は、PCST[2:0]にはSTLが出力される。

【００９１】（例６）デバッグモードからノーマルモードへの移行時のＰＣトレース

図１６にデバッグモードからの復帰時のPCST[2:0]の出力タイミングを示す。デバッグ例外、デバッグリセットからの復帰命令DERET命令の分岐ディレイスロット命令までがデバッグモードに属する。DERET命令の戻り先の命令からノーマルモードになり、ＰＣトレースが有効になる。

【００９２】＜トレーストリガの詳細説明＞PCST[2:0]信号へのトレーストリガの出力について説明する。命令アドレストレーストリガ、データアドレストレーストリガ、プロセッサバストレーストリガのいずれかが発生した場合、次のような論理でトレーストリガ情報がPCST[2:0]に出力される。

（１）その時に分岐成立の分岐命令もしくはジャンプ命令が実行されているか、例外が発生している場合

ここにおいてもしトレーストリガが発生しなければ、PCST[2:0]にはJMP、BRT、EXPコードが出力されるはずである。この（１）の場合、にはトレーストリガが発生してもPCST[2:0]の出力は変化せずに保留され、直後の（２）または（３）のケースで、トレーストリガ情報が出力される。

（２）その時にパイプラインがストールしている場合

ここにおいてもしトレーストリガが発生しなければ、PCST[2:0]には、STLコードが出力されるはずである。この（２）の場合、トレーストリガが発生すると、PCST[2:0]にはTSTコードが出力される。

（３）（１）及び（２）以外の場合、すなわちパイプラインが順次実行をしている場合

ここにおいてもしトレーストリガが発生しなければ、PCST[2:0]にはSEQコードが出力されるはずである。この（３）の場合、トレーストリガが発生すると、PCST[2:0]にはTSQコードが出力される。

【００９３】以下に実際の例を波形図にて示す。

【００９４】（例１）トレーストリガの発生例

図１７に、通常の命令を順次実行中のトレーストリガの発生例を示す。add命令実行中にトレーストリガが発生

したので、トレーストリガのゴードTSQを出力する。

【００９５】（例２）例外発生命令実行中にトレーストリガが発生した場合の例

図１８に、例外発生命令実行中にトレーストリガが発生した例を示す。ソフトウエアブレーク命令break実行中にトレーストリガが発生したが、break命令実行のクロックではPCST[2:0]信号に例外発生のコードEXPを出力し、次のクロックでトレーストリガのコードを出力する。この例ではストール状態なので、TSTコードを出力する。

【００９６】（例３）インダイレクトジャンプ命令実行中にトレーストリガが発生した場合の例

図１９に、インダイレクトジャンプ命令実行中にトレーストリガが発生した場合の例を示す。インダイレクトジャンプ命令jr2を実行中にトレーストリガが発生したが、jr2命令実行のクロックではPCST[2:0]信号にジャンプのコードJMPを出力し、次のクロックでトレーストリガのコードを出力する。この例では命令実行状態なので、TSQコードを出力する。

【００９７】＜トレースメモリインターフェース回路の説明＞図２０に、デバッグツール６０内のトレースメモリインターフェース６６０とトレースメモリ６７０の構成図を示す。

【００９８】デバッグモジュール６０から出力されたTPC及びPCST[2:0]は、トレースデータレジスタ６６３を通してトレースメモリ６７０へ書き込まれる。この時のアドレスの値は、トレースアドレスカウンタ６６２から供給される。PCST[2:0]はトレーストリガデコーダ６６４にも入力されており、トレーストリガの発生をトレースメモリコントロール回路６６１へ伝える。トレースアドレスカウンタの初期値の設定、インクリメント／停止の指示は、トレーストリガデコーダ６６４の出力結果に基づいてトレースメモリコントロール回路６６１が行なう。

【００９９】コントローラ６３０がトレースメモリ６７０の内容を読み出す場合には、コントローラアドレスレジスにアドレスを設定してトレースメモリコントロール回路６６１にリード要求を出すと、データがコントローラデータレジスタ６６７に書き込まれるので、これを読み出す。

【０１００】コントローラ６３０がトレースメモリ６７０へデータを書き込む場合には、コントローラアドレスレジス６６５にアドレスを設定し、コントローラデータレジスタ６６７にデータを設定し、トレースメモリコントロール回路６６１にライト要求を出す。

【０１０１】＜低消費電力化の詳細説明＞図２１に、デバッグモジュール内の電源供給の構造を示す。

【０１０２】デバッグツールが接続されていない場合、信号DBGE*はハイレベルになっている。ユーザリセット時にこのDBGE*信号がハイレベルの場合には、電源供給スイッチ３８が切断され、シリアルモニタバス回路３４、ＰＣトレース回路３２には、電源が供給されなくなる。これらの回路は、外部デバッグツールが接続されていない時には使用する意味がないので、電源を供給しないことにより、マイクロプロセッサ全体での消費電力を低減することができる。

【０１０３】デバッグツールが接続されていない場合でも、命令／データアドレスブレーク回路３１、プロセッサバスブレーク回路３３及びレジスタ回路３５には電源が供給され、命令／データアドレスブレークとプロセッサバスブレークの機能は機能する。デバッグ例外のベクタアドレスをユーザ領域に変更することにより、これらのハードウエアブレーク機能をユーザがデバッグの用途に使用できる。

【０１０４】＜デバッグモジュール初期化回路の詳細説明＞図２２に、デバッグモジュール初期化回路を示す。

【０１０５】デバッグモジュールが接続されていない時にはDBGE*がハイレベルになるので、ユーザリセットがアサートされると、デバッグモジュール初期化信号がアサートされ、デバッグモジュール３０が初期化される。

【０１０６】デバッグツールが接続されていない場合でも、電源ラインへのノイズなどが原因でデバッグモジュール自体が誤動作してしまい、プロセッサコア２０へ割込みを要求したりする可能性がある。デバッグツールは接続されていないので、デバッグリセット(DRESET*)信号を入れることはできない。ユーザリセットではデバッグモジュールを初期化できないと、このデバッグモジュールを初期化する手段がない。従って、デバッグツールが接続されていない場合に、ユーザリセットがデバッグモジュールを初期化することは極めて重要である。

【０１０７】逆に、デバッグツールが接続されている場合には、DBGE*がローレベルになるので、ユーザリセットがアサートされても、デバッグモジュール初期化信号はアサートされず、デバッグモジュール３０が初期化されない。

【０１０８】この動作は、ユーザリセットの直後にトレーストリガをかけたい場合などに極めて有効である。もしユーザリセットでデバッグモジュール３０が初期化されてしまうと、例えばトレーストリガをかけるためにはいったんデバッグモードに入って必要なレジスタを設定し直さなければならない。しかし、デバッグモードに入ることで、その間、実時間動作ができなくなってしまうので、とらえようとしている現象がとらえられない可能性が高い。

【０１０９】電源ラインへのノイズなどが原因でデバッグモジュール自体が誤動作してしまった場合でも、デバッグツールが接続してある場合、デバッグツールからのデバッグリセット(DRESET*)で初期化できるので問題ない。

【０１１０】

【効果】以上詳細に説明したように、本発明によれば、従来例に比べて、次のような効果が得られる。

【０１１１】従来例１に比べて、次のような効果が得られる。

・モニタプログラムをデバッグツール側のメモリで走らせるため、ユーザのメモリを使用しない。

・モニタへ入るために専用のデバッグ例外を用意しているため、ユーザの割込みに制限を加えない。

・ユーザターゲットシステムにシリアルインターフェースを用意する必要がない。

・ハードウエアブレークポイントが使用可能である。

【０１１２】従来例２に比べて、次のような効果が得られる。

・マイクロプロセッサ内部にシーケンサをもつ必要がないため、マイクロプロセッサ内部に追加されるデバッグのためのロジック回路が簡単である。

・モニタプログラムによって、レジスタにアクセスするため、派生品のマイクロプロセッサでレジスタが追加されるような場合でも、モニタを変更するだけで容易にそれらに対するアクセスが可能になる。

・上記２項の理由により、マイクロプロセッサコアに多種類の周辺回路を付け加えれている場合ても、デバッグのためのロジック回路を共通化できる。この共通モジュールを周辺回路の一部としてマイクロプロセッサに内蔵することで、プロセッサコアが共通で周辺回路が異なる多様な派生マイクロプロセッサに対して共通のデバッグツールを適用することが可能となる。

【０１１３】従来例３に比べて、次のような効果が得られる。

・デバッグツールのハードウエア仕様を共通化できる。

・デバッグツールとの接続のための信号の本数を減らせる。

・上記理由により、プローブを小型化、低価格化できる。

・ユーザターゲット上のマイクロプロセッサがメモリやＩ／Ｏをアクセスするので、デバッグツールに要求されるタイミング条件が改善される。

・デバッグツールに接続されない信号に関しては、影響を及ぼさない。

・デバッグツールとマイクロプロセッサ間の通信速度を必要に応じて、遅くできる。

・上記理由により、高速のマイクロプロセッサにも対応できる。

【０１１４】従来例４に比べて、次のような効果が得られる。

・デバッグツールを用いて、ターゲットメモリやＩ／Ｏへのアクセス及び実行制御が実現される。

・キャッシュメモリで実行された命令についても、そのアドレス情報がトレースできる。

【図面の簡単な説明】

【図１】従来例１の構成を示す図。

【図２】従来例２の構成を示す図。

【図３】従来例３の構成を示す図。

【図４】従来例４の構成を示す図。

【図５】本発明の一実施例の構成を示す図。

【図６】本発明の一実施例中のデバッグモジュールの構成を示す図。

【図７】本発明の一実施例中のデバッグツールの構成を示す図。

【図８】本発明の一実施例中のシリアルモニタバスに関わる回路の概略図を示す図。

【図９】本願の一実施例中のシリアルモニタバスのライトバスオペレーションのタイミングチャート。

【図１０】本願の一実施例中のシリアルモニタバスのライトバスオペレーションのタイミングチャート。

【図１１】本願の一実施例における分岐命令のＰＣトレース出力の例を示すタイミングチャート。

【図１２】本願の一実施例におけるインダイレクトジャンプ命令のＰＣトレース出力の例を示すタイミングチャート。

【図１３】本願の一実施例における例外とインダイレクトジャンプ命令のＰＣトレース出力の例を示すタイミングチャート。

【図１４】本願の一実施例におけるデバッグ例外発生時のPCST[2:0]の出力タイミングの例を示すタイミングチャート。

【図１５】本願の一実施例において、デバッグ例外発生時にターゲットＰＣを出力している場合のPCST[2:0]出力タイミングを示すタイミングチャート。

【図１６】本願の一実施例におけるデバッグモードからの復帰時のPCST[2:0]の出力タイミングを示すタイミングチャート。

【図１７】通常の命令を順次実行中のトレーストリガの発生の例のタイミングチャート。

【図１８】例外発生命令実行中にトレーストリガが発生した場合の例のタイミングチャート。

【図１９】インダイレクトジャンプ命令実行中にトレーストリガが発生した場合の例のタイミングチャート。

【図２０】デバッグツール６０内のトレースメモリインターフェースとトレースメモリの構成を示す図。

【図２１】デバッグモジュール内の電源供給部の構成を示す図。

【図２２】デバッグモジュール初期化回路の構成を示す図。

【符号の説明】

１０：マイクロプロセッサ

１５：内部デバッグインターフェース

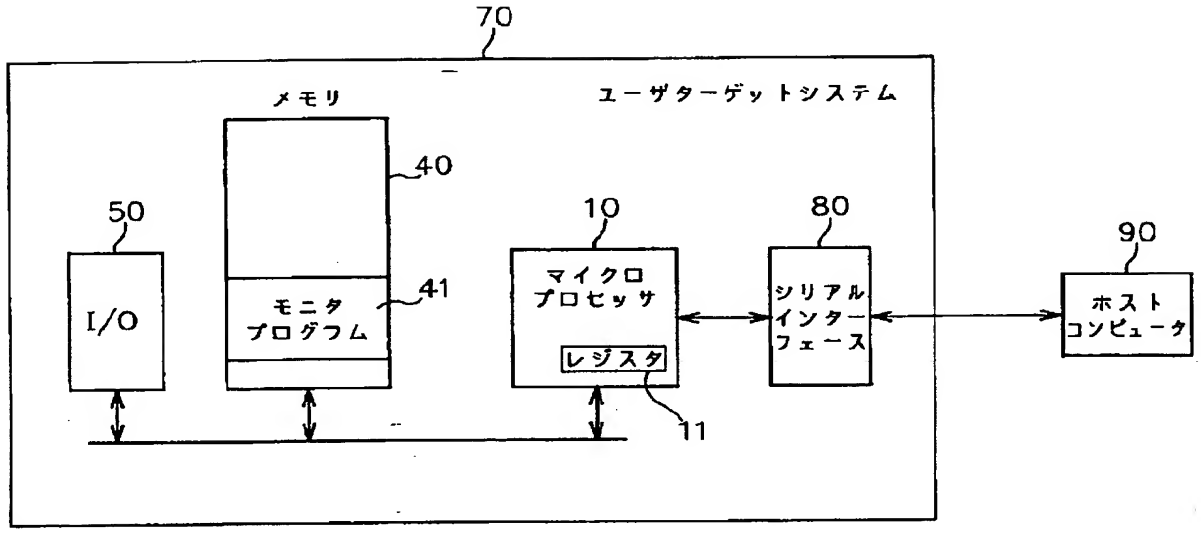
１６：内部プロセッサバス

２０：プロセッサコア

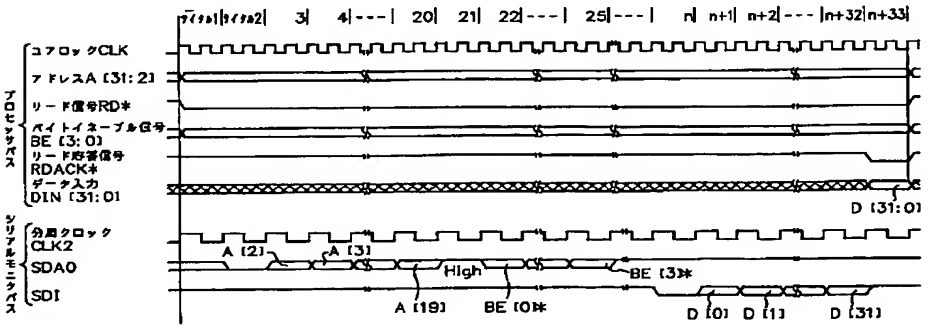
３０：デバッグモジュール

３１：命令／データアドレスブレーク回路
３２：ＰＣトレース回路
３３：プロセッサバスブレーク回路
３４：シリアルモニタバス回路
３５：レジスタ回路
３６：外部インターフェース回路
３７：分周回路
３８：電源供給スイッチ
４０：メモリ
５０：Ｉ／Ｏ
６０：デバッグツール
７０：ユーザターゲットシステム
７１：外部デバッグインターフェース
９０：プロセッサバス
３４２：シリアル出力回路Ａ
３４３：シリアル入力回路Ａ
６２０：通信インターフェース
６３０：コントローラ
６４０：モニタメモリ
６５０：モニタメモリインターフェース
６５１：シリアル入力回路Ｂ
６５２：シリアル出力回路Ｂ
６６０：トレースメモリインターフェース
６６１：トレースメモリコントロール回路
６６２：トレースアドレスカウンタ
６６３：トレースデータレジスタ
６６４：トレーストリガデコータ
６６５：コントローラアドレスレジスタ
６６７：コントローラデータレジスタ
６７０：トレースメモリ
６８０：ランコントロール
６９０：ターゲットインターフェース

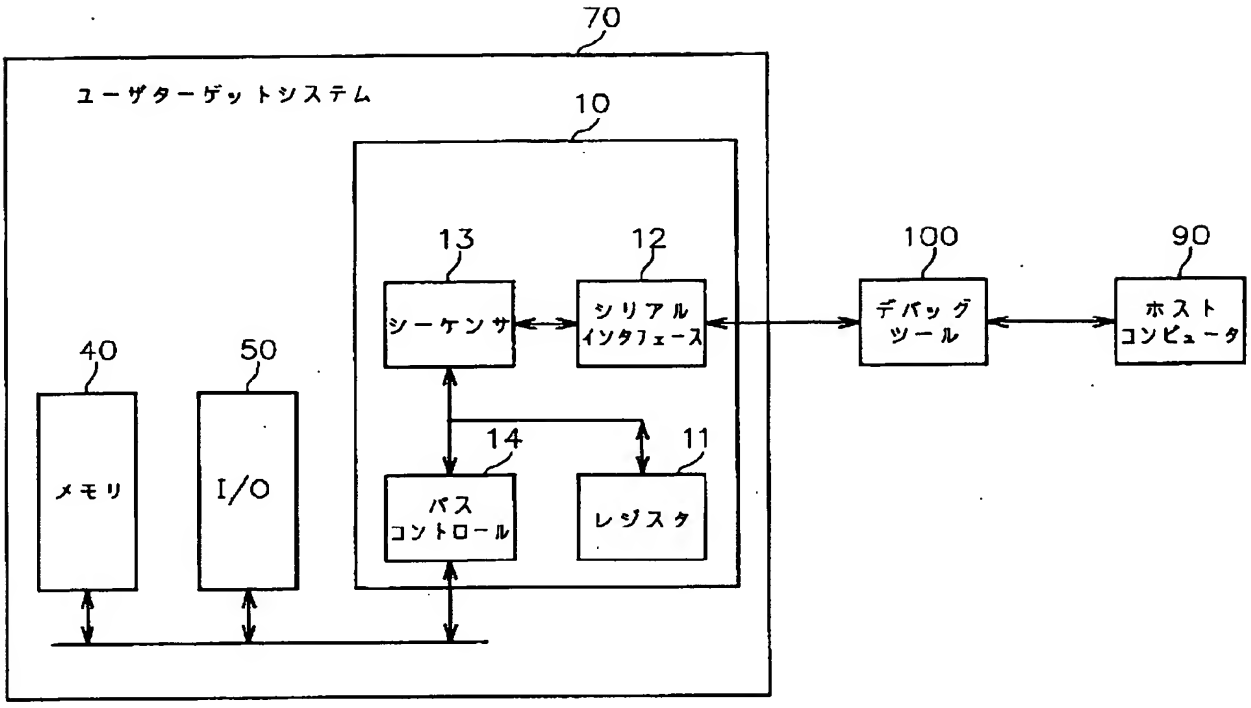
【図１】



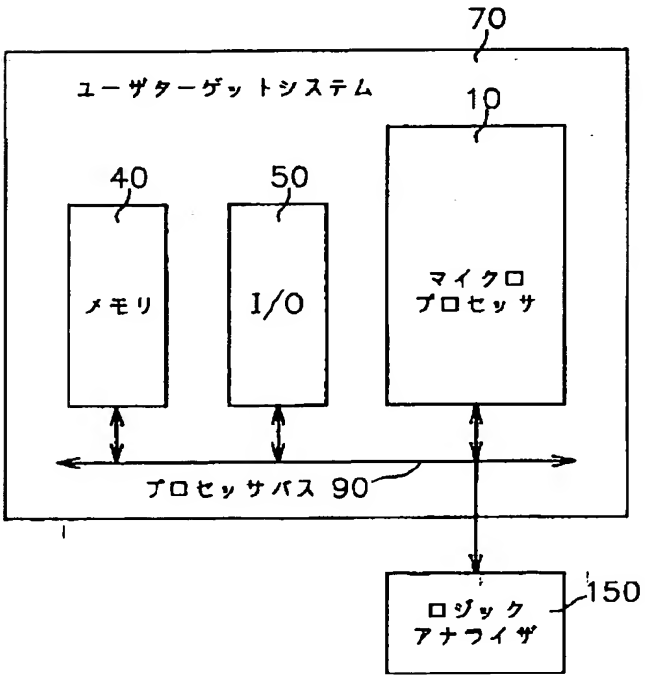
【図９】



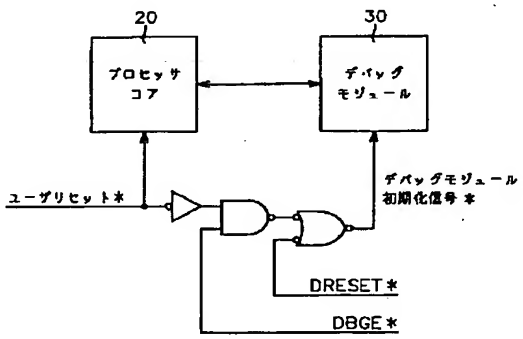
【図２】


70
ユーザターゲットシステム
10
13
12
100
90
シーケンサ
シリアル
インタフェース
デバッグ
ツール
ホスト
コンピュータ
40
50
メモリ
I/O
14
11
バス
コントロール
レジスタ

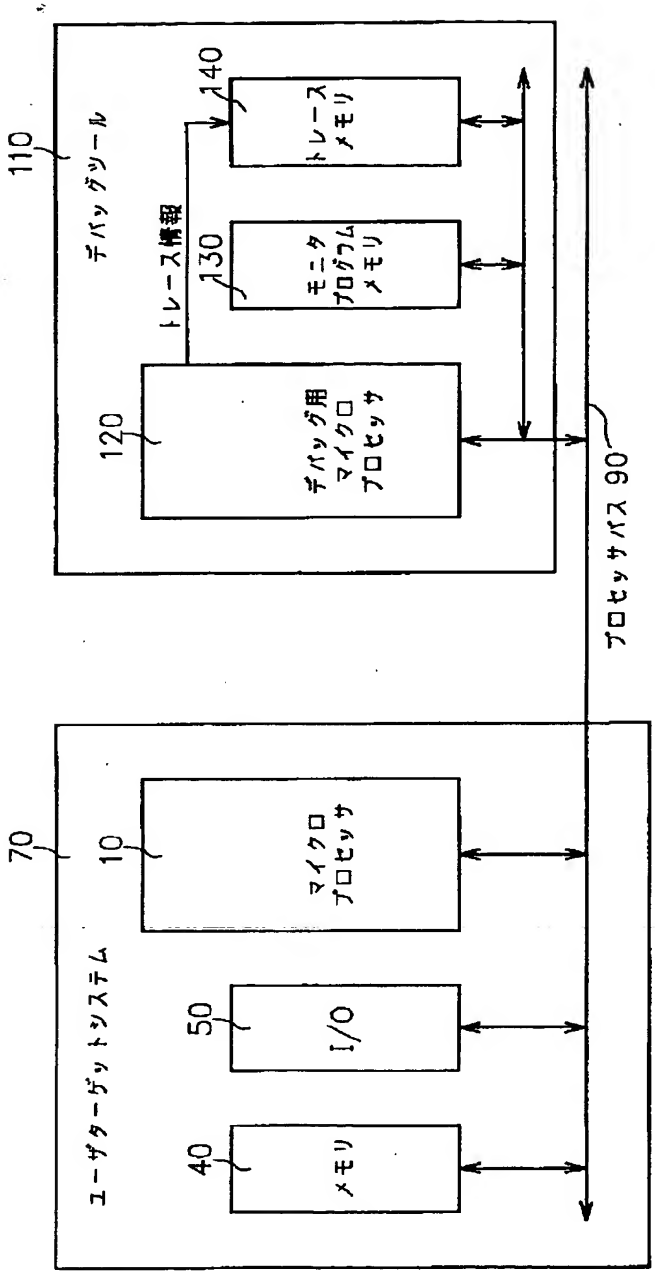
【図４】


70
ユーザターゲットシステム
10
40
50
メモリ
I/O
マイクロ
プロセッサ
プロセッサバス 90
ロジック
アナライザ
150

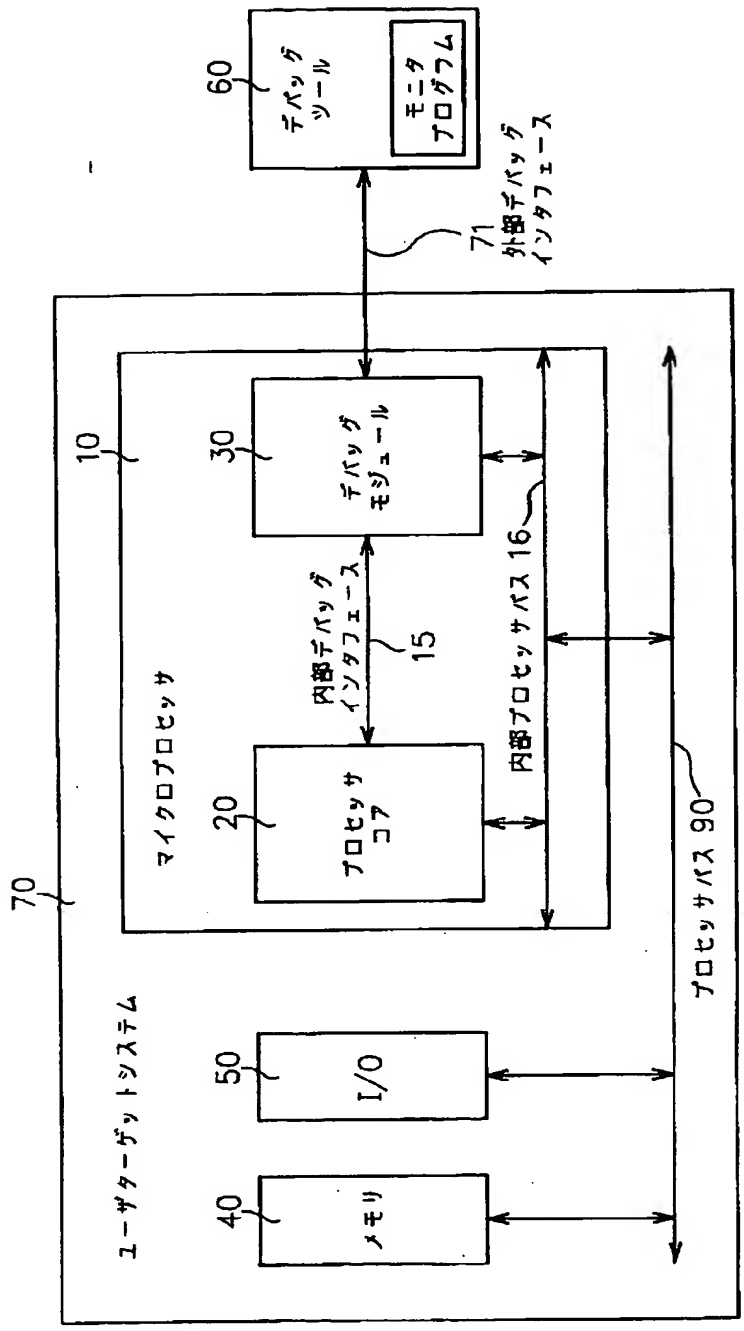
【図２２】


20
30
プロセッサ
コア
デバッグ
モジュール
ユーザリセット＊
デバッグモジュール
初期化信号＊
DRESET＊
DBGE＊

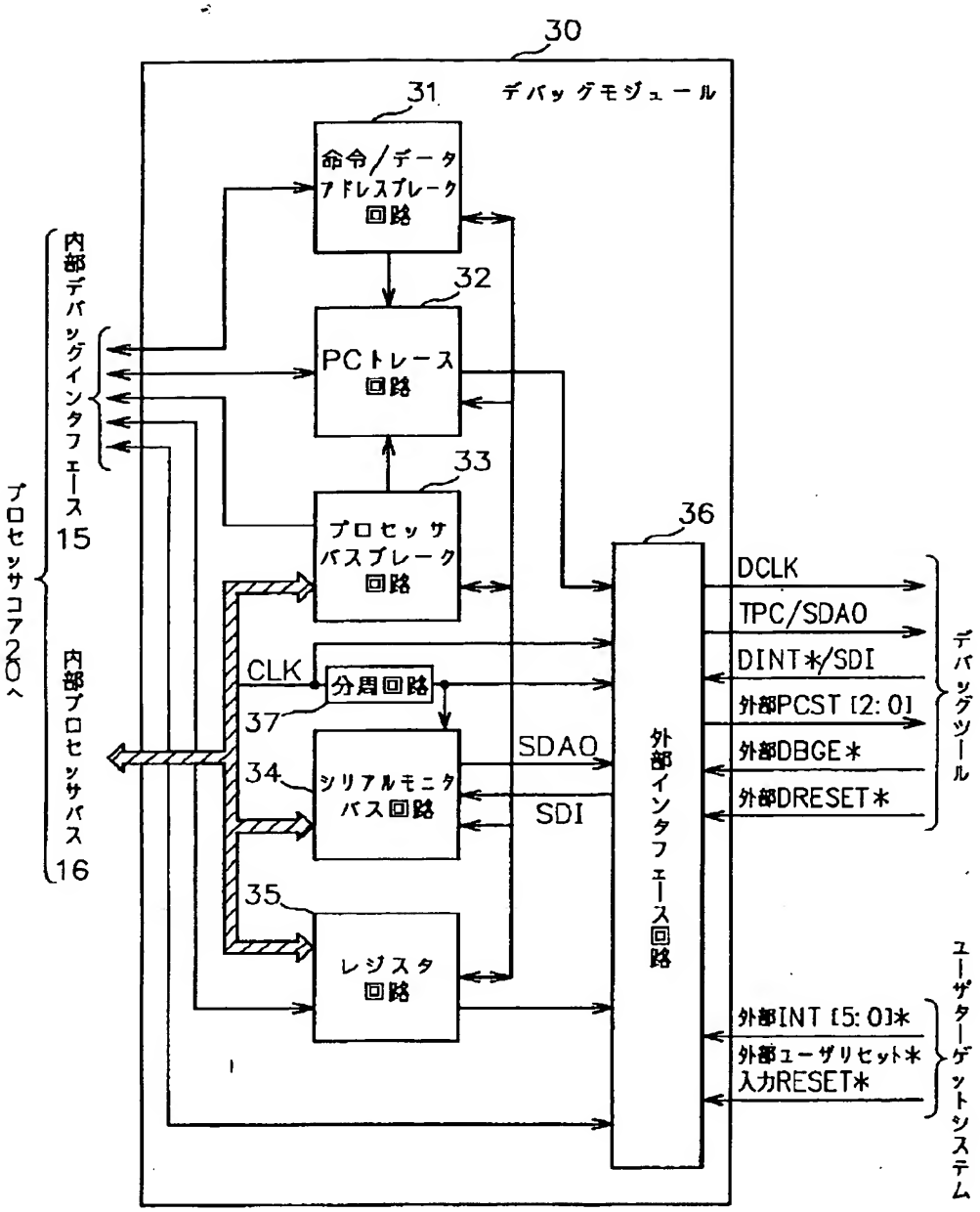
【図３】


110
デバッグツール
140
トレース
メモリ
130
モニタ
プログラム
メモリ
トレース情報
120
デバッグ用
マイクロ
プロセッサ
プロセッサバス 90
70
ユーザターゲットシステム
10
マイクロ
プロセッサ
50
I/O
40
メモリ

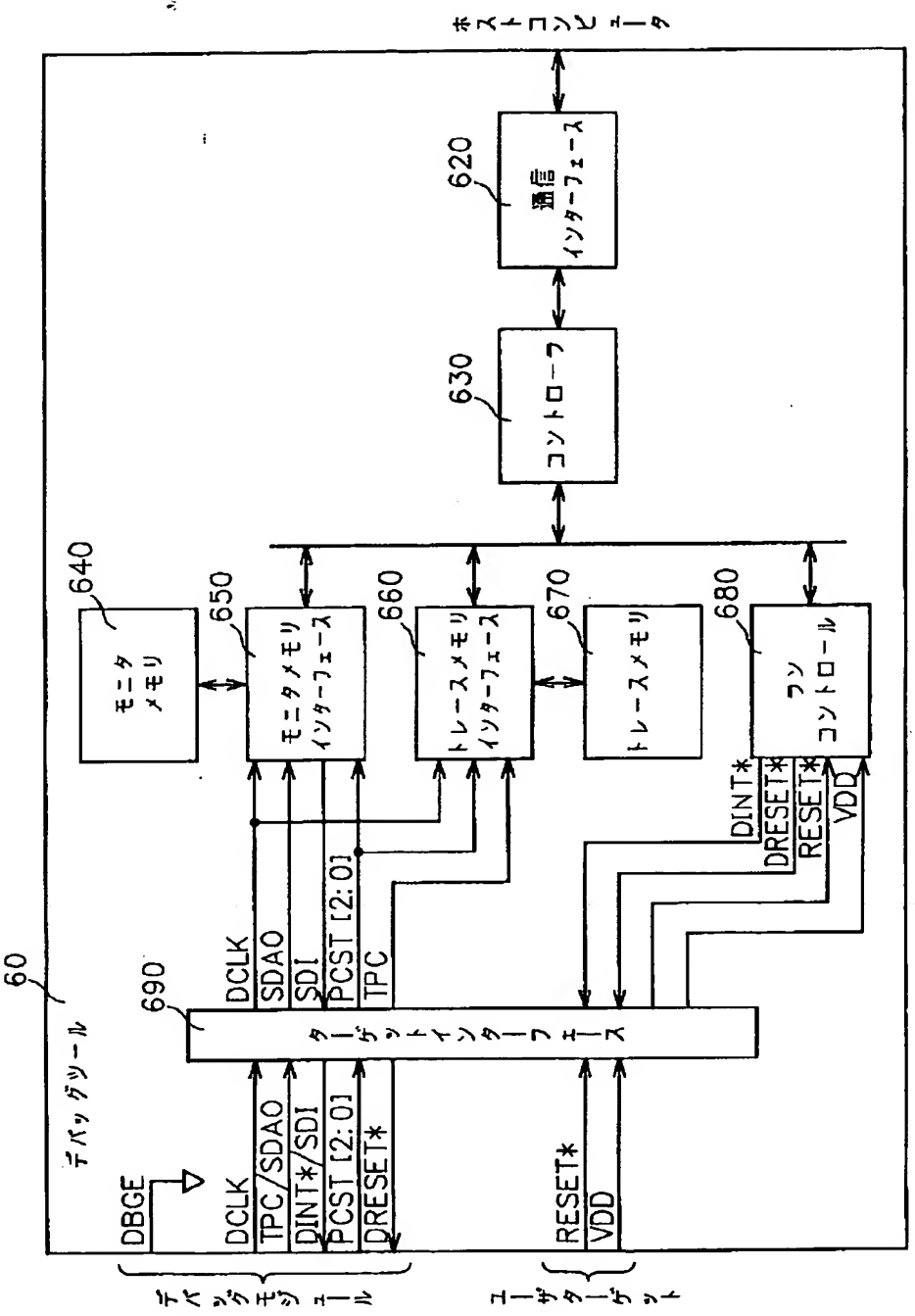
【図５】


60
デバッグ
ツール
モニタ
プログラム
71
外部デバッグ
インタフェース
70
10
マイクロプロセッサ
30
デバッグ
モジュール
内部デバッグ
インタフェース
15
内部プロセッサバス 16
20
プロセッサ
コア
プロセッサバス 90
ユーザターゲットシステム
50
I/O
40
メモリ

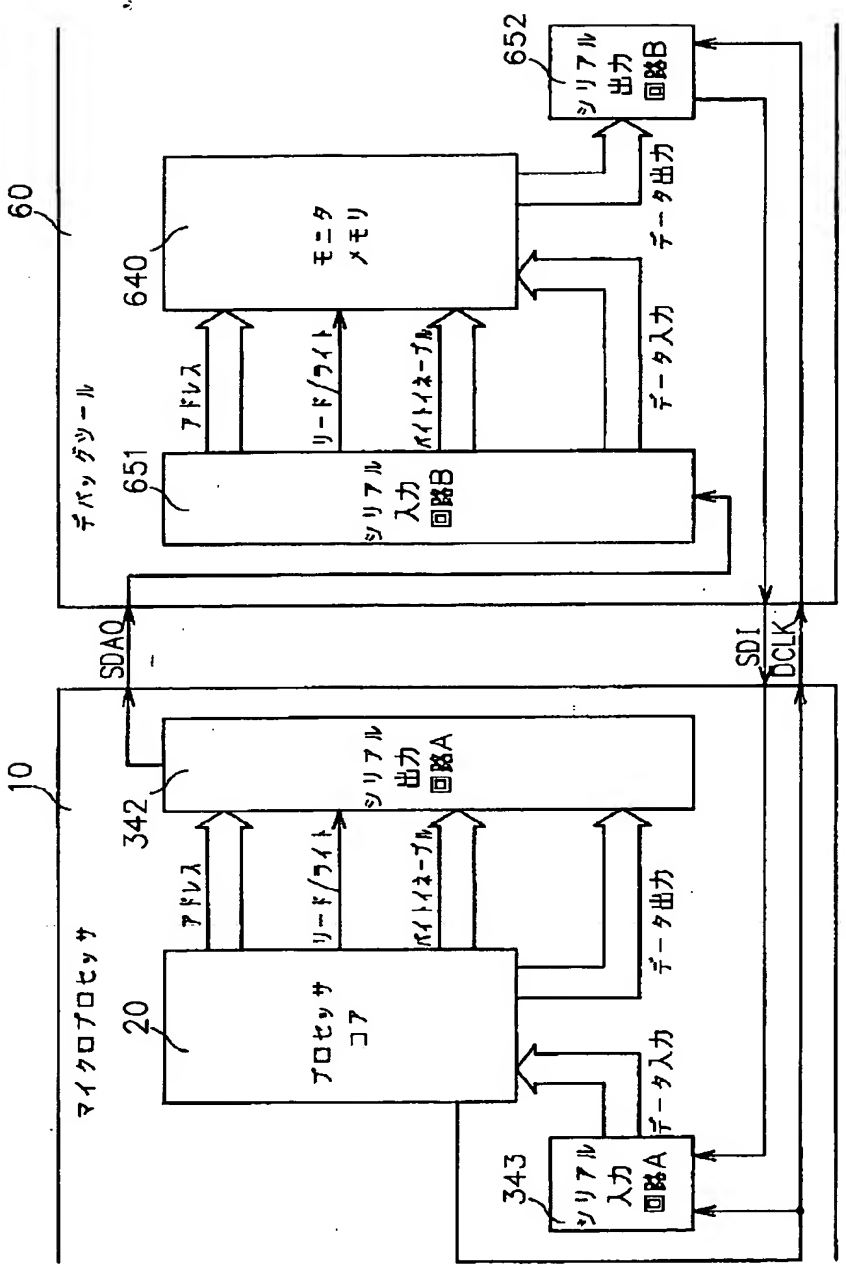
【図６】


30
デバッグモジュール
31
命令/データアドレスブレーク回路
32
PCトレース回路
33
プロセッサバスブレーク回路
36
分周回路
37
CLK
34
シリアルモニタバス回路
35
レジスタ回路
SDAO
SDI
外部インタフェース回路
DCLK
TPC/SDAO
DINT*/SDI
外部PCST [2:0]
外部DBGE*
外部DRESET*
デバッグツール
外部INT [5:0]*
外部ユーザリセット*
入力RESET*
ユーザターゲットシステム
内部デバッグインタフェース 15
内部プロセッサバス 16
プロセッサコア20へ

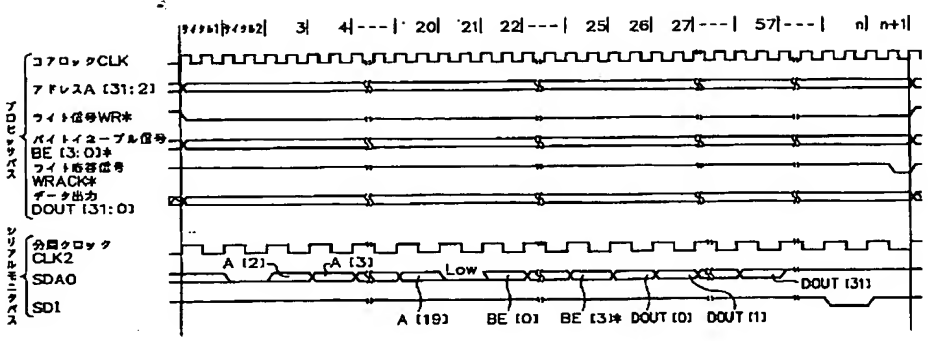
【図７】


ホストコンピュータ
620
通信インターフェース
630
コントローラ
640
モニタメモリ
650
モニタメモリインターフェース
660
トレースメモリインターフェース
670
トレースメモリ
680
コントロール
DINT*
DRESET*
RESET*
VDD
60
デバッグツール
690
ターゲットインターフェース
DCLK
SDAO
SDI
PCST [2:0]
TPC
DBGE
DCLK
TPC/SDAO
DINT*/SDI
PCST [2:0]
DRESET*
RESET*
VDD
デバッグモジュール
ユーザターゲット

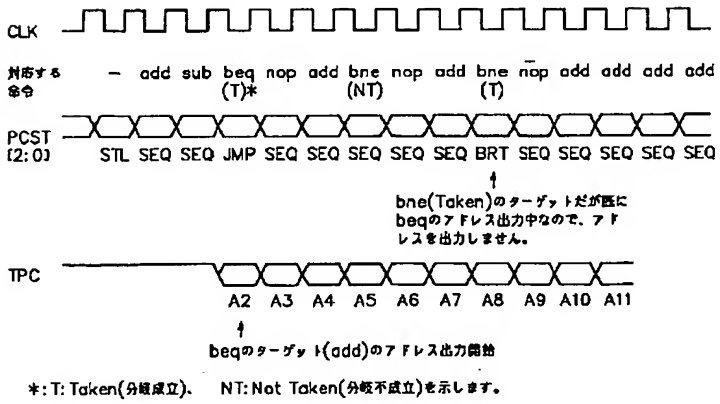
【図8】


10
マイクロプロセッサ
20
プロセッサ コア
342
シリアル 出力 回路A
343
シリアル 入力 回路A
アドレス
リード/ライト
バイトイネーブル
データ出力
データ入力
60
デバッグツール
640
モニタ メモリ
651
シリアル 入力 回路B
652
シリアル 出力 回路B
SDAO
SDI
DCLK

【図１０】

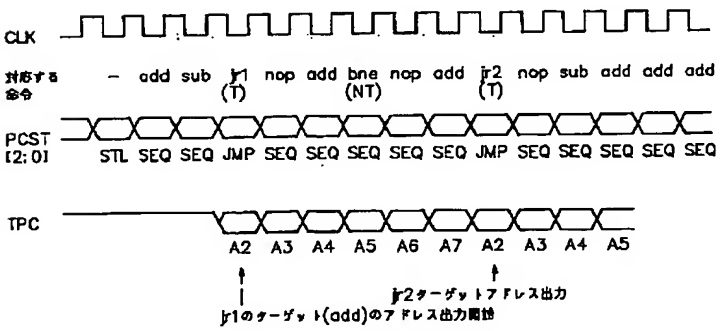


【図１１】

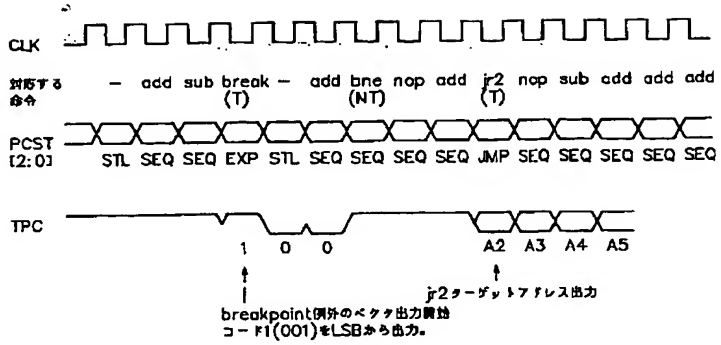


*: T: Taken(分岐成立)、 NT: Not Taken(分岐不成立)を示します。

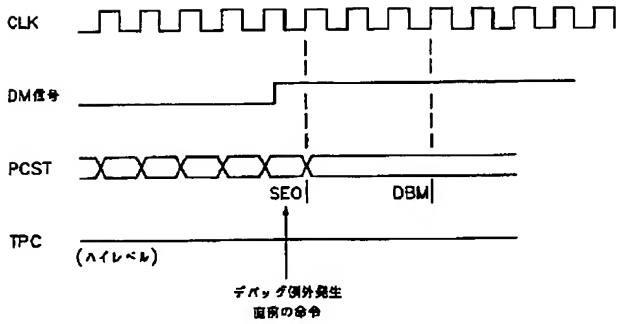
【図１２】



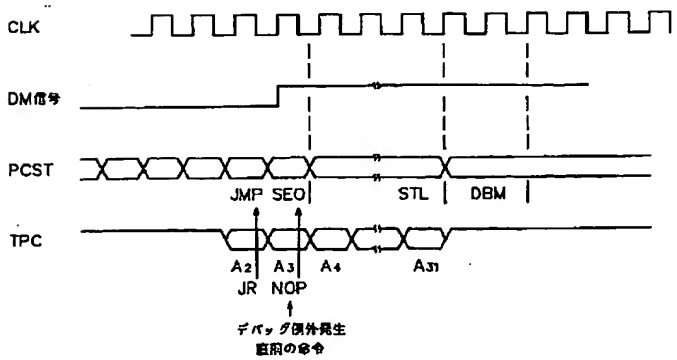
【図１３】


CLK
対応する命令
– add sub break – add bne nop add jr2 nop sub add add add
(T) (NT) (T)
PCST [2:0]
STL SEQ SEQ EXP STL SEQ SEQ SEQ SEQ JMP SEQ SEQ SEQ SEQ SEQ
TPC
1 0 0
A2 A3 A4 A5
breakpoint例外のベクタ出力開始コード1(001)をLSBから出力。
jr2ターゲットアドレス出力

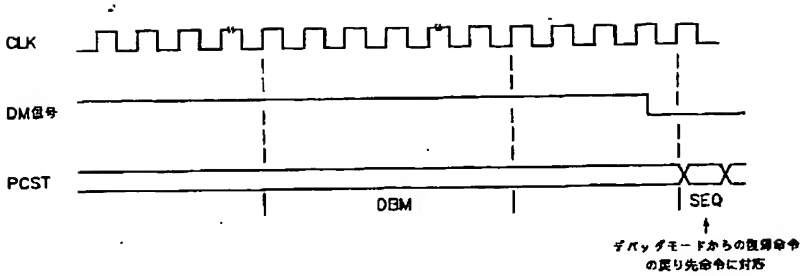
【図１４】


CLK
DM信号
PCST
SEQ
DBM
TPC
(ハイレベル)
デバッグ例外発生直前の命令

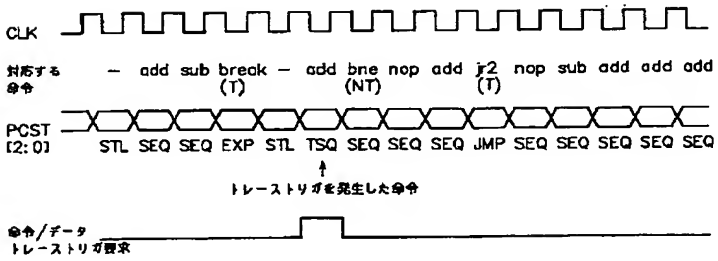
【図１５】


CLK
DM信号
PCST
JMP SEQ
STL
DBM
TPC
A2 A3 A4 A31
JR NOP
デバッグ例外発生直前の命令

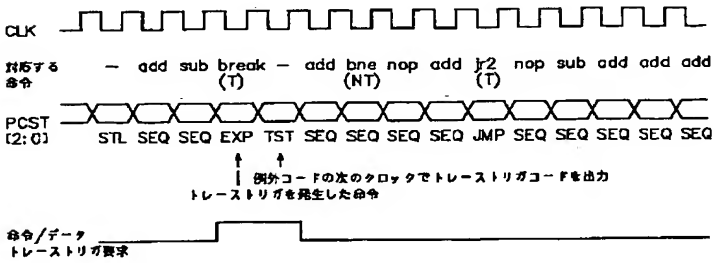
【図１６】



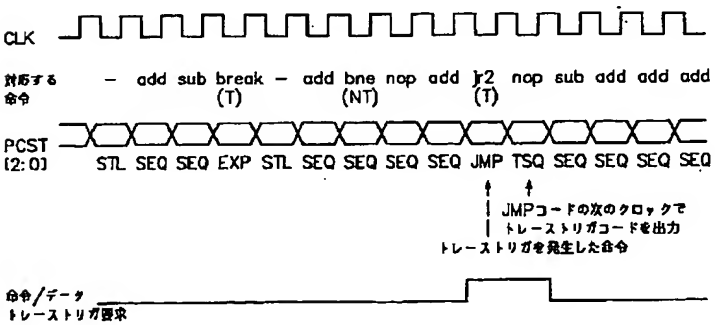
【図１７】



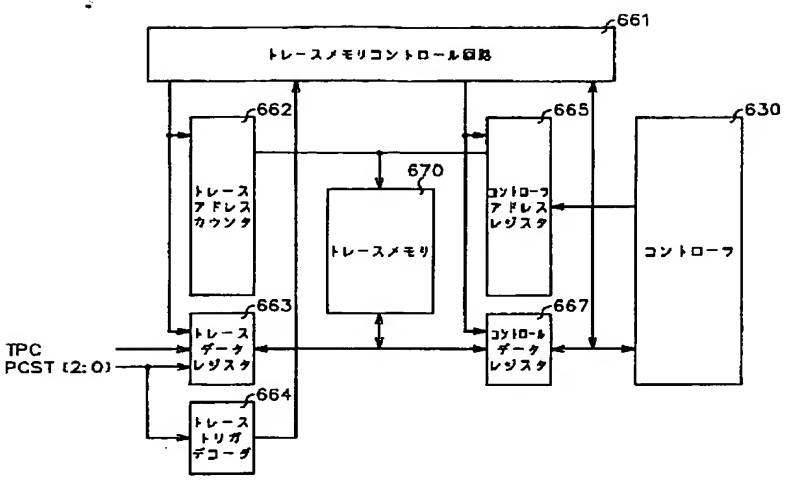
【図１８】



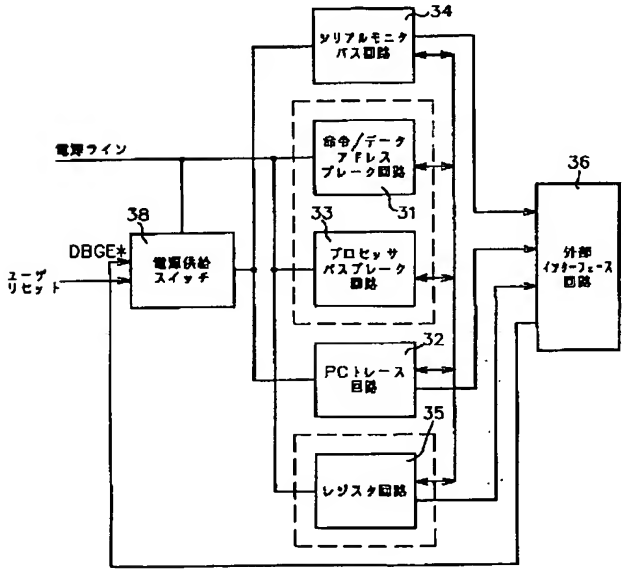
【図１９】



【図２０】


661
トレースメモリコントロール回路
662
トレース
アドレス
カウンタ
665
630
670
コントローラ
アドレス
レジスタ
トレースメモリ
コントローラ
663
667
TPC
PCST〔2:0〕
トレース
データ
レジスタ
コントロール
データ
レジスタ
664
トレース
トリガ
デコーダ

【図２１】


34
シリアルモニタ
バス回路
電源ライン
命令/データ
アドレス
ブレーク回路
36
31
38
33
DBGE*
電源供給
スイッチ
プロセッサ
バスブレーク
回路
外部
インターフェース
回路
ユーザ
リセット
32
PCトレース
回路
35
レジスタ回路